MobiEyes: Distributed Architecture for Location-based Services

Ling Liu

Georgia Institute of Technology

Jointly with Buğra Gedik, Kipp Jones, Anand Murugappan, Bhuvan Bamba

Outline of the Talk

- Motivation
- MobiEyes' Distributed Architecture
  - Design Ideas
  - Sever Side Optimization
  - Mobile Client Side Optimization
- Evaluation
- Location Privacy
  - MobiEyes' approach – Personalized *k* Anonimization

Location Based Services

- "Location-based" services: what are they?
  - Services based on the location of a principal
  - allow customers or applications to request and receive information based on their geographic locations
    - maps, location dependent activities, emergency response, law enforcement, inventory control, geo-fencing, demographic data collection, and so on.
- Technological drivers:
  - Cell phones equipped with WiFi, Bluetooth, GPS;
  - Telematics, RFID tags, DHCP and IEEE 802.11 (Wireless LAN)
- Growing field:
  - U.S. wireless location-based services revenues will be exceeding $4 billion in the U.S. and $30 billion worldwide in 2005.

[Source: Kevira, Inc.]

Example Location Based Applications

- Location based services:
  - *location-dependent information delivery service*
    - Electronic tour guides (*ex*: CyberGuide)
    - Transportation guides (*ex*: NextBus)
    - Buddy trackers
  - *location-aware emergency services*
    - Roadside assistance (*ex*: NetworkCar)
    - General emergency (*ex:* FCC's Phase II E911
  - *location-based advertisement*
    - *mCommerce applications*
  - *location-enhanced entertainment*
    - Mobile games (*ex*: *Mogi*)

Location Queries

- A location query is a *moving query (MQ) over mobile moving objects*
  - expressed as a *spatial continuous moving query over locations of mobile objects*
- Components of a MQ
  - Focal object (mobile or still)
  - Spatial Region (moving or static)
  - Query Filter
- Result of a MQ:
  - Objects that are inside the spatial area covered by the location query's spatial constraint (region) and satisfy the query filter
  - Target Objects (mobile or still)

Example Location Queries

- Moving queries over moving objects (the most general form of location queries)
  - Finding all my buddies within 10 miles on highway 85 North every 10 minutes in the next 5 hours
  - “Give me the positions of those *customers who are looking for a taxi* and are *within 5 miles*, during next 20 minutes”
- Moving queries over static objects
  - “Give me the locations and names of the gas stations that are

Characteristics of Moving queries over moving objects

- Target objects of a location query changes continuously as the focal object or the objects being queried move continuously
- Spatial region of a location query is continuously changing as the focal object moves continuously.

LBSs: Problems and Assumptions

- Location is dynamically changing information
  - Location-dependent computing
- Cost of communication is asymmetric
  - Broadcast v.s. point to point communication
- Severe power restrictions on mobile hosts
  - Power constraints
- Limited resource available on mobile hosts
  - Computing Resource constraints
- Frequent and foreseeable disconnections
  - Service continuity
- Security issues due to mobility of hosts
  - Location security + location privacy

Assumptions

- Moving objects are able to *locate their positions*
- Moving objects have *computational capabilities* to carry out arbitrary computational tasks
- The geographical area of interest is covered by several *base stations*, which are connected to a central server or a server farm
- The communication is *asymmetric*
- Moving objects have *synchronized clocks*

Location-based Service Architecture Alternatives

- Key Functionality
  - Location query processing (moving queries over moving objects)
- Centralized client-server architecture
  - Mobile clients report their positions periodically;
  - Servers handle the location query and location data management.
- Distributed client-server architecture
  - Partition the location query processing task into server site processing and mobile object side processing;
  - Using server mediation to establish the communication between mobile objects.
- Decentralized peer to peer architecture
  - Mobile clients serve as server, client, and router for each location query and location data management task;
  - LBS system does not have central control or knowledge of all nodes.

LBSs: State of Art

- Most of existing LBSs use centralized architecture
- Known Assumptions
  - Large number of location queries over a fixed set of mobile objects
- Technical Focus in literature
  - Server side optimization
    - Spatial-Tempo indexing techniques based on multidimensional indexes, such as R-Tree, kd-tree, Grid file
    - Selective broadcast scheduling algorithms

MobiEyes: Problem Statement

- How to handle increasing number of mobile objects with relatively smaller number of location queries?

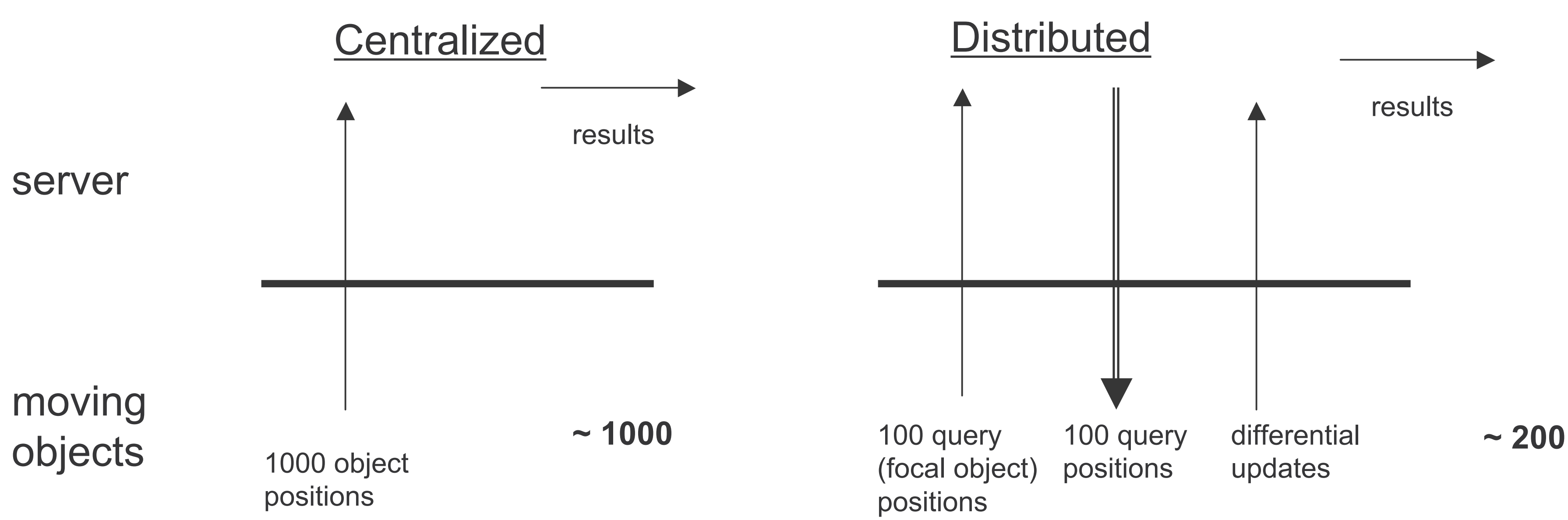
- How do we evaluate moving location queries (MQs) over moving objects ***efficiently***, in order to reduce or minimize
  - Server Load
  - (Wireless) Communication Bandwidth
  - Amount of Computation on Mobile Objects

Important observation

- Moving location queries are location dependent.
  - Only those mobile objects that are in the geographical vicinity of the focal objects of active location queries are relevant.
- Possible solutions
  - **Location-dependent indexing at the server side**
    - Motion-adaptive indexing, Broadcast optimization [[CIKM 2004, ACM GIS 2004, TKDE 2006]
    - But the bandwidth required for location updates is still high
  - **Geographical partitioning of location queries based on their spatial validity scope can be effective.**
    - It ships some location query processing to the relevant mobile clients.
    - Only those mobile objects that are relevant to some active location queries will report their location updates
    - [EDBT2004, ACM TMC 2005]

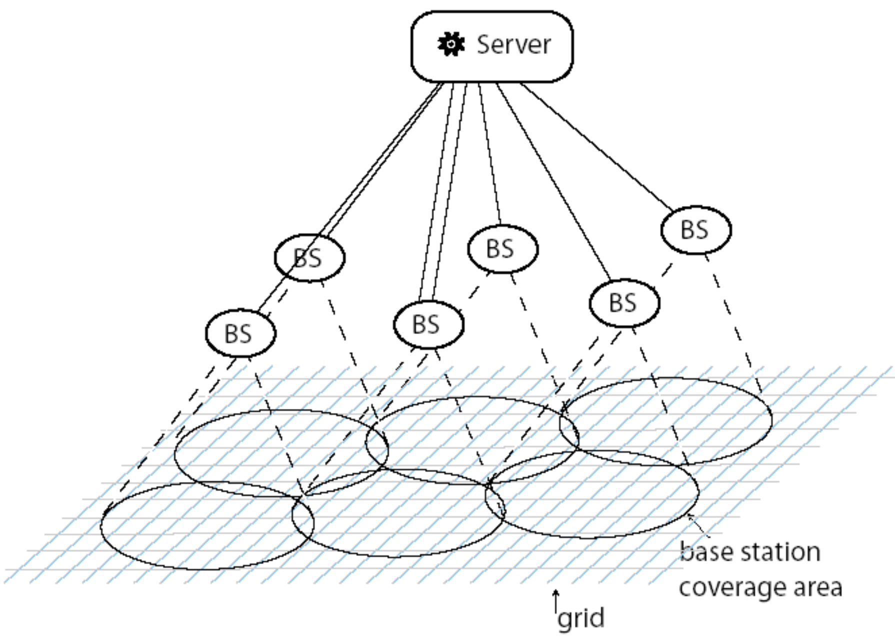
A Simple Scenario

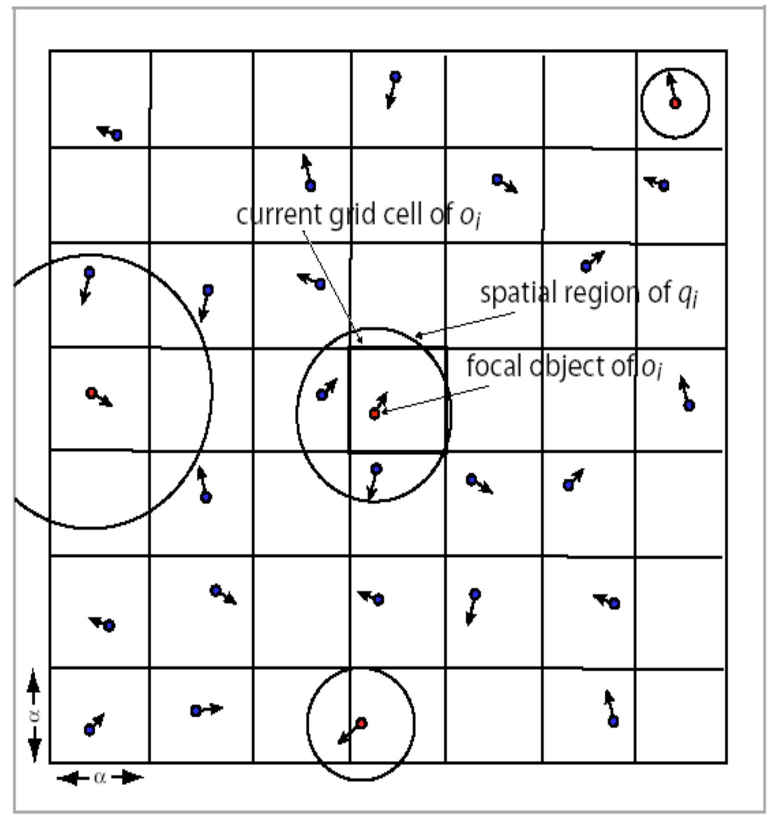
- 1000 mobile objects, 100 location queries



use velocity vectors instead of positions

System Model





MobiEyes: Distributed Architecture

- **Servers**
  - Register and maintain all location service requests (queries)
  - mediate the processing of location based service requests among mobile clients and between a mobile client and the server
- **Mobile clients**
  - dynamically track if a moving object is entering (leaving) some query regions defined by the nearby moving queries;
  - report only important location changes to the server periodically or aperiodically;
  - share location information and communicate with one another through server mediation/
- **Important Performance Consideration**
  - Server load (scalability), and network bandwidth
  - Scalable partition of client and server responsibilities using localization schemes
  - energy consumption at mobile clients

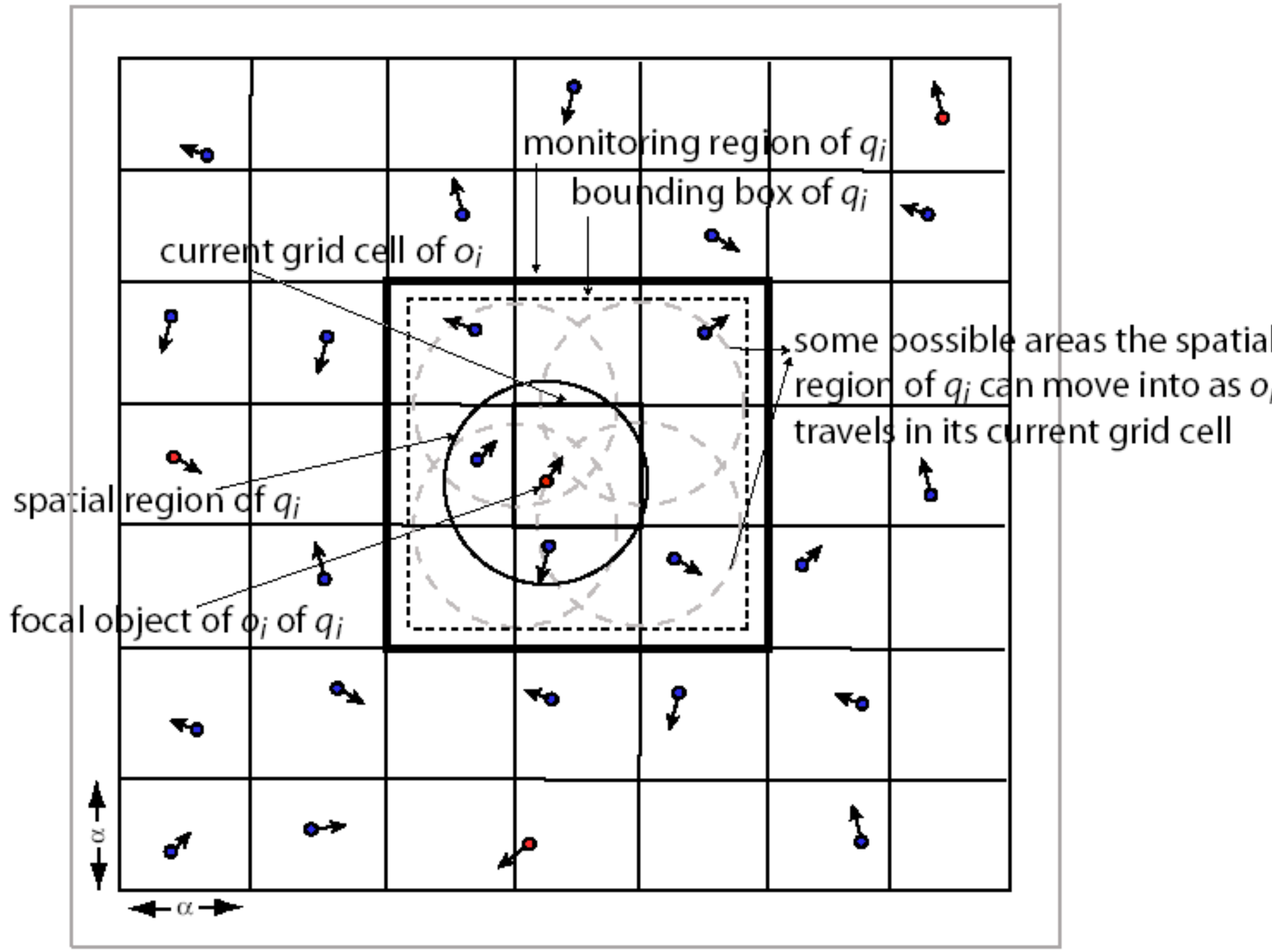
Basic Key Concepts and Illustrations

Grid and Grid cells

Current Grid Cell of an Object

Bounding Box of a Query

Monitoring Region of a Query



Why is MobiEyes solution interesting?

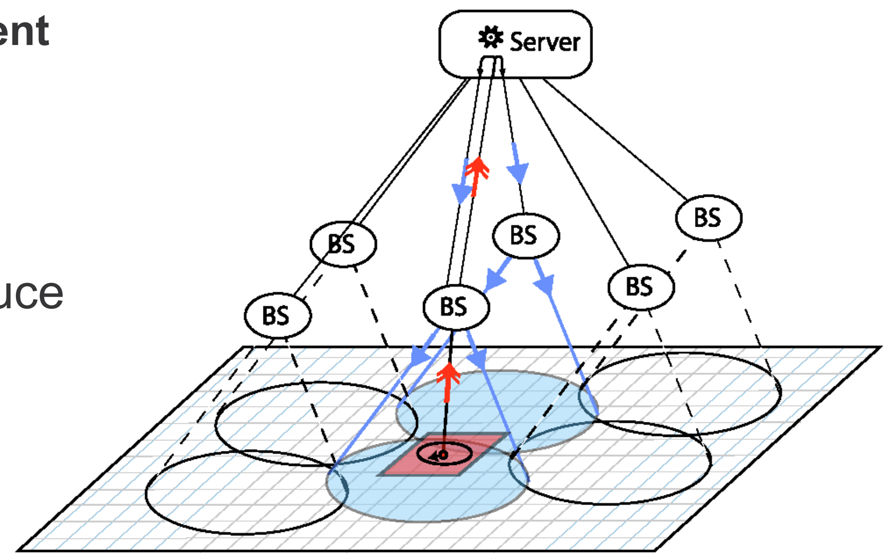
- ***Moving computation close to places where data is produced***
  - The location and the dynamic attributes of the moving objects, which are of interest to the queries, are remote to the server *but* they are local to the moving objects.
- ***Perform computation to save communication***
  - The computational capabilities of the mobile objects can be utilized in a distributed solution, to decrease the load on the server and increase scalability.
  - Moreover the additional processing on the moving object side can be utilized to decrease communication.
- ***Exploit communication asymmetry***
  - Although the dynamic nature of the system requires updates on the query states to be conveyed to a possibly large number of moving objects, the communication asymmetry inherent in mobile communications makes it efficient to convey this information to appropriate moving objects.

MobiEyes: Installing Queries

- Installation of a query into the system is composed of two phases.
  - First, the server state should be updated to reflect the installation of the query.
  - Second, the query should be propagated to and installed on the right set of moving objects (within its monitoring region).
    - Find the base stations that cover the monitoring region of the query
    - Broadcast the query using these base stations
    - Objects that receive the broadcast install the query if they locate inside the monitoring region of the query

MobiEyes Optimization Techniques

- **Reduce the communication from moving objects to the server**
  - by only **reporting velocity changes**
  - By reporting **the focal object position** changes when they **move out of its current grid cell.**
- Mobile Object Side Query Processing
- Mobile Object Side Optimization
  - Using **dead reckoning** to further reduce the amount of reporting on velocity
  - Handling Grid Cell Changes
  - **Safe Period Optimization**
  - **Lazy** v.s. Eager **Query Propagation**
- **Moving query Grouping**
  - Reducing the redundant processing steps at both server and the moving objects side.



Mobile Object Site
Query Processing Logic

- A moving object periodically processes all queries in its *LQT* table.
- For each query, it *predicts the position of the focal object of the query* using the velocity, time and position information available in the *LQT* entry of the query
- Then by comparing its current position and the *estimated* position of the query's focal object, it *determines whether itself is covered by the query's spatial region*.
- In case *the result is different from the last result* computed in the previous time step, the object *notifies the server* of this change

LQT

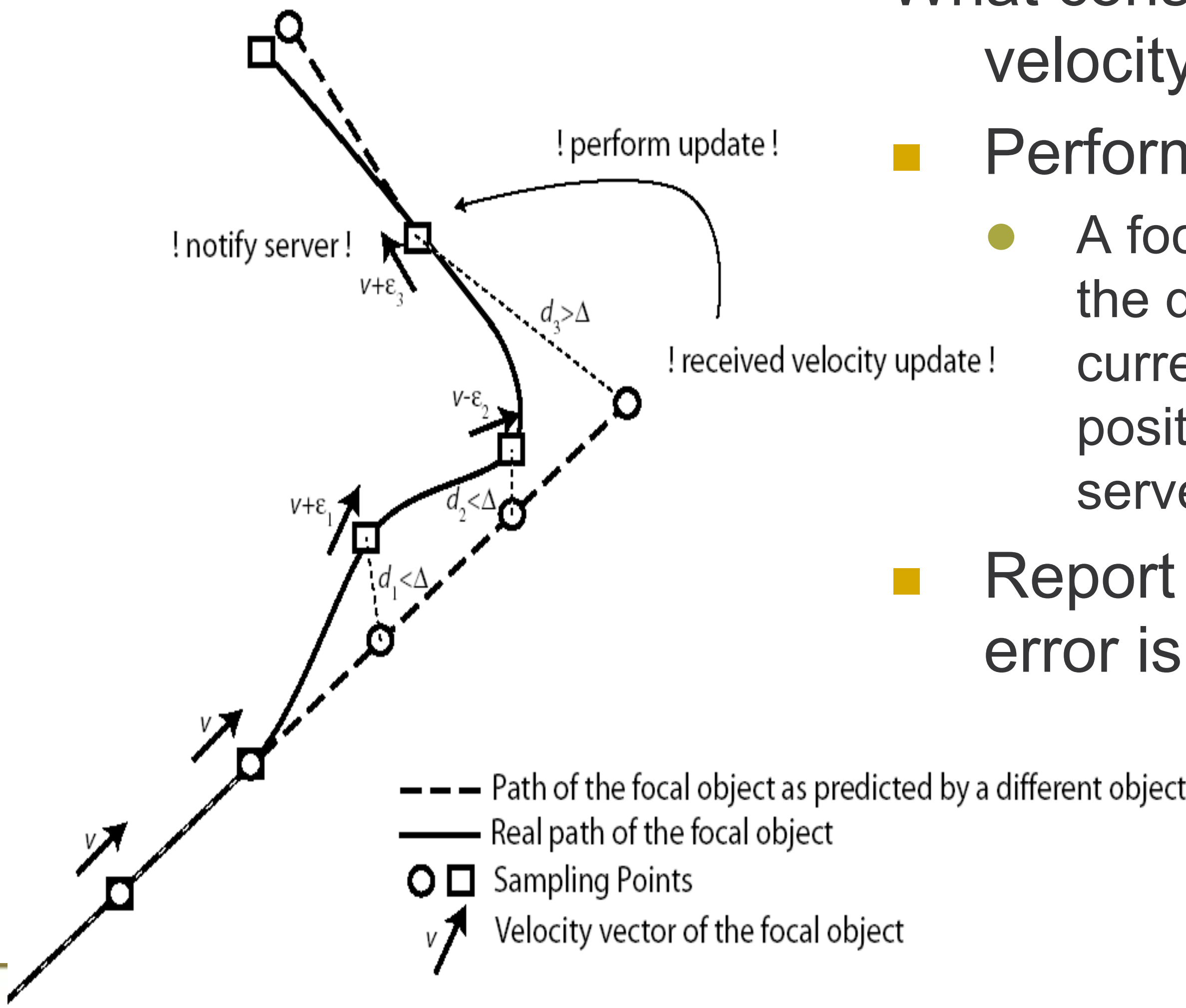
scan

qid	pos	vel	tm	region	.	inOut
q_i	p_i	v_i	t_i	circle(0,0,*r*)	.	$inout_i$

Let *ct* be current time
Let *pos* be my current position
$fpos = p_i + vel*(ct - t_i)$
IF *pos* in circle(*fpos*.x,*fpos*.*y*, *r*)
  THEN we are in
    IF $inout_i$ == *true*
      THEN do nothing
      ELSE notify the server regarding my
        inclusion in the query result
  ELSE we are out
    // dual processing here

Georgia Institute of Technology
Georgia Tech College of Computing

Velocity Change Estimation: Dead Reckoning

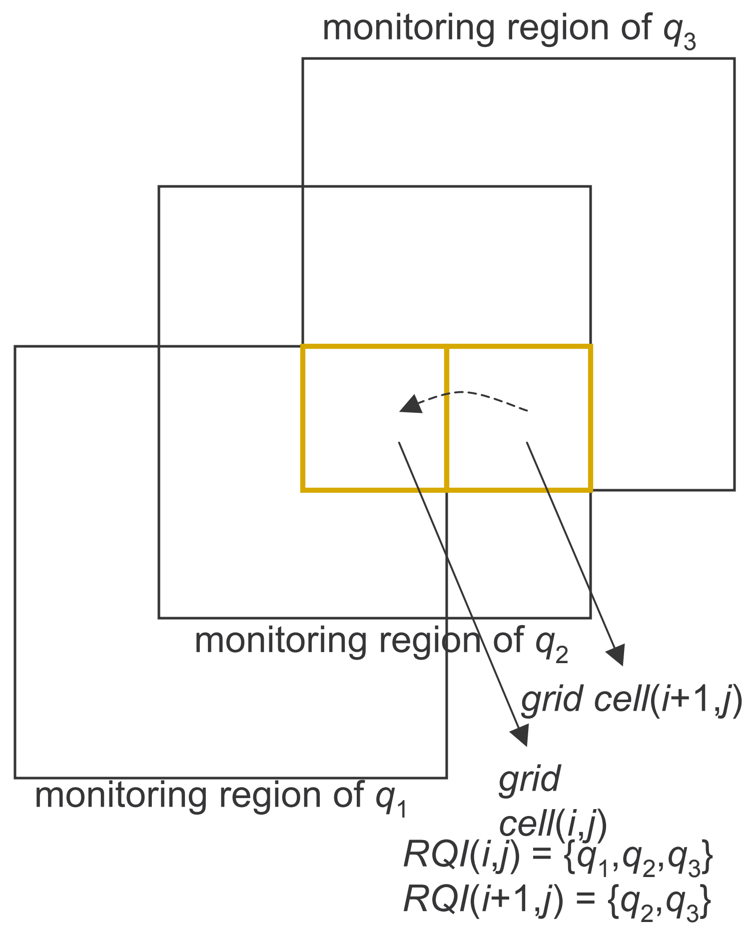


What constitutes a significant velocity change?

- Perform dead reckoning
  - A focal object calculates the difference between its current position and its last position broadcasted by the server.
- Report only when the error is > Δ

Handling Grid Cell Changes

- Focal object changes its current grid cell
  - Need to contact the server to remove/add/update queries on the moving objects
- Non-focal object changes its current grid cell
  - Need to contact the server
    - receive new queries
    - update/remove existing queries
  - Immediate Propagation v.s. Lazy Propagation



Lazy Query Propagation

- **Advantage:**
  - Eliminate the need for non-focal objects to contact the server when they change their grid cells
  - Trading off some accuracy.
- **Mechanism**
  - Instead of receiving the new queries from the server and installing them immediately, an object can *wait until velocity vector or cell change notifications* regarding the focal objects of these queries are broadcasted to the area in which the object locates.
  - The velocity vector change notifications are expanded to include the spatial region and the filter of the queries
  - The object installs the new queries when it receives the velocity vector change broadcast.
- **Tradeoff:**
  - The object will be unaware of the query until a velocity vector (or grid cell) change of the query occurs, introducing some inaccuracy.

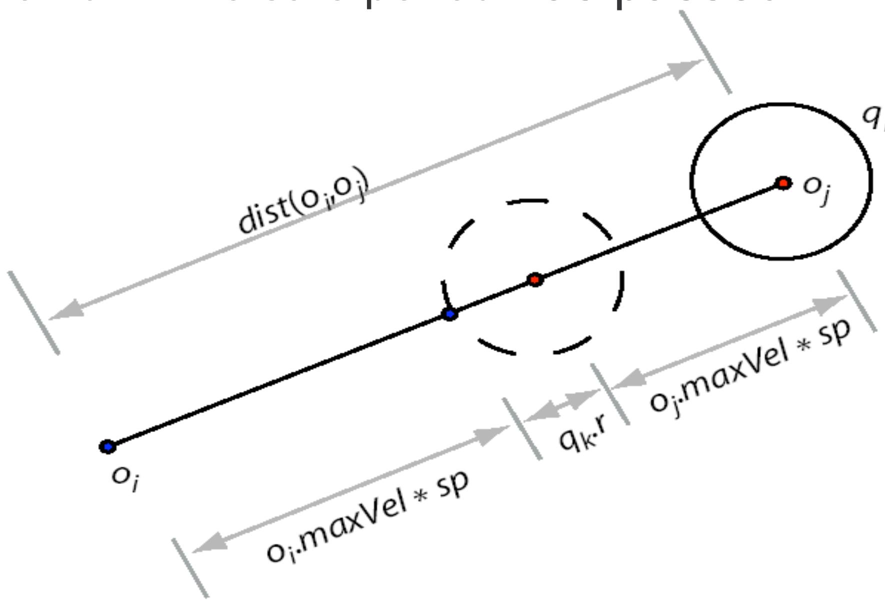
Optimizations: Safe Periods

Assumption: the maximum velocities of objects are known

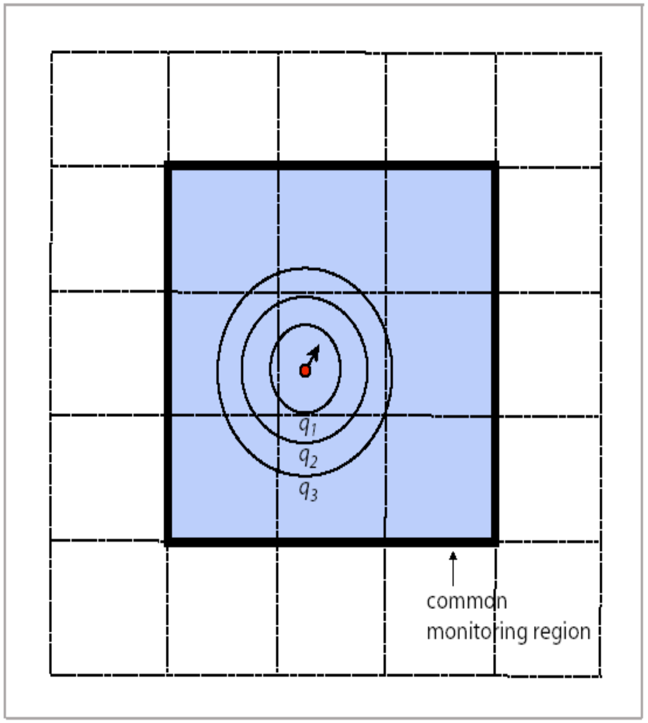
- The *safe period* (*sp*) of the object with respect to the query Q
  - The time period needed for an object to be located inside the monitoring region of Q
- An optimization to speed up the query processing on the moving object side.
  - For each query in its LQT table, an object can calculate a *worst case lower bound on the amount of time that has to pass* for the object to locate inside the monitoring area of the query.

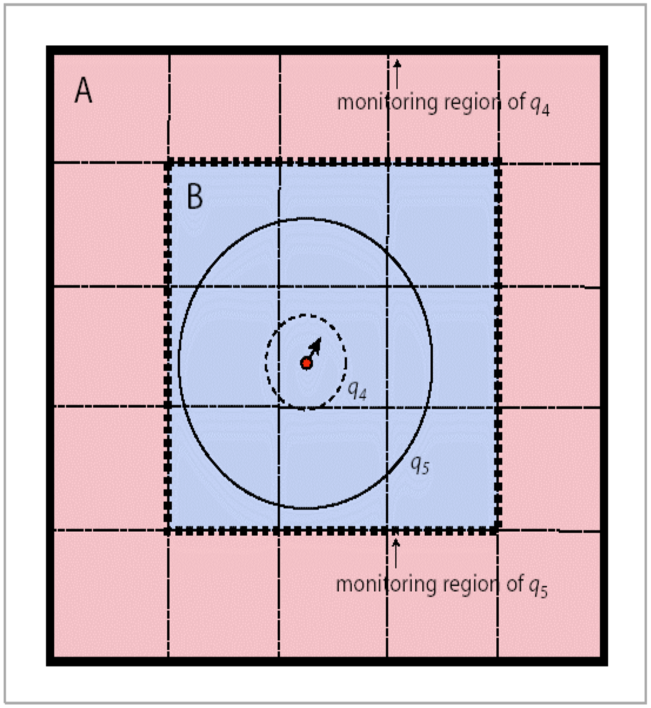
Advantage:

- Once the safe period sp is calculated for a query, it is safe to skip that query's periodic evaluation until the safe period has passed.



Optimizations: Grouping





full grouping
Grouping queries with
the matching monitoring regions

partial grouping
Grouping queries with the same
focal objects but non-matching
monitoring regions

Experiments

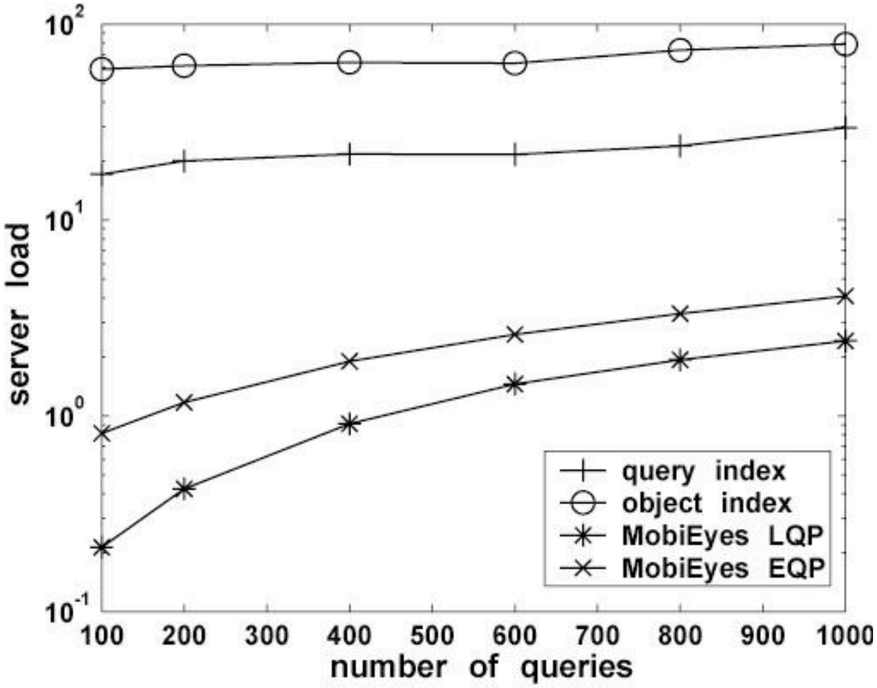
- **Measures:**
  - Server load
  - Messaging cost
  - Amount of processing on moving objects

Parameter	Description	Value range	Default value
ts	Time step	30 seconds	
α	Grid cell side length	0.5-16 miles	5 miles
no	Number of objects	1,000-10,000	10,000
nmq	Number of moving queries	100-1,000	1,000
nmo	Number of objects changing velocity vector per time step	100-1,000	1,000
$area$	Area of consideration	100,000 square miles	
$alen$	Base station side length	5-80 miles	10 miles
$qradius$	Query radius	{3, 2, 1, 4, 5} miles	
$qselect$	Query selectivity	0.75	
$mospeed$	Max. object speed	{100, 50, 150, 200, 250} miles/hour	

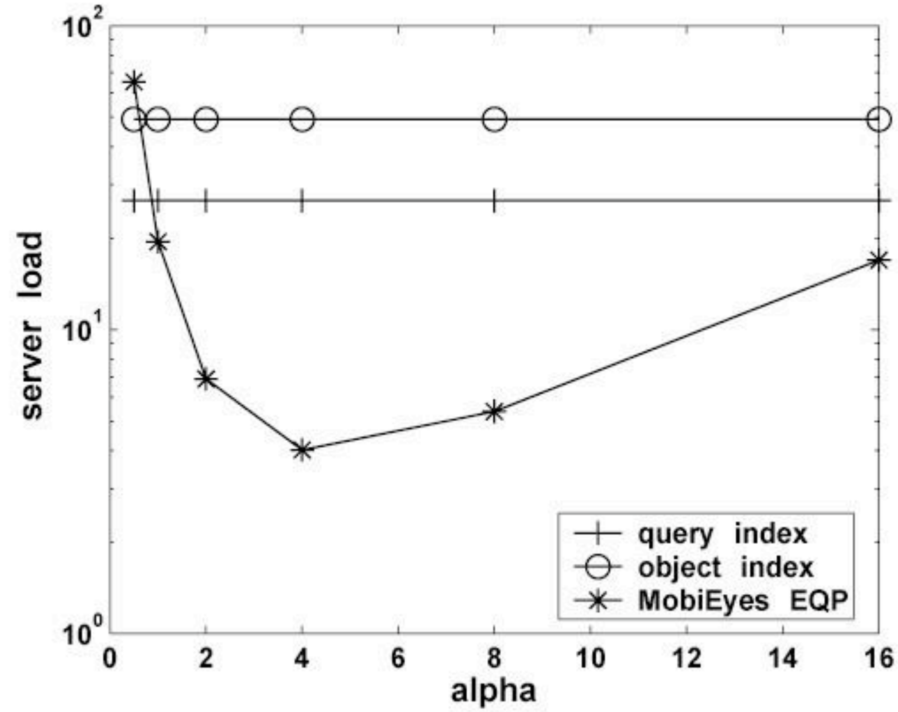
Two Basic Centralized Server side solutions

- **Object index** (naïve)
  - Build an R*-tree on object positions
  - Update the index on every position change
  - Periodically Re-evaluate each query using the index
- **Query index**
  - Build an R*-tree on query positions
  - Update the index on every focal object position change (upon arrival of new position)
    - Identify queries affected by the new position update
    - Add or remove the object from the queries identified
      - Allow differential update of the query result

Server Load



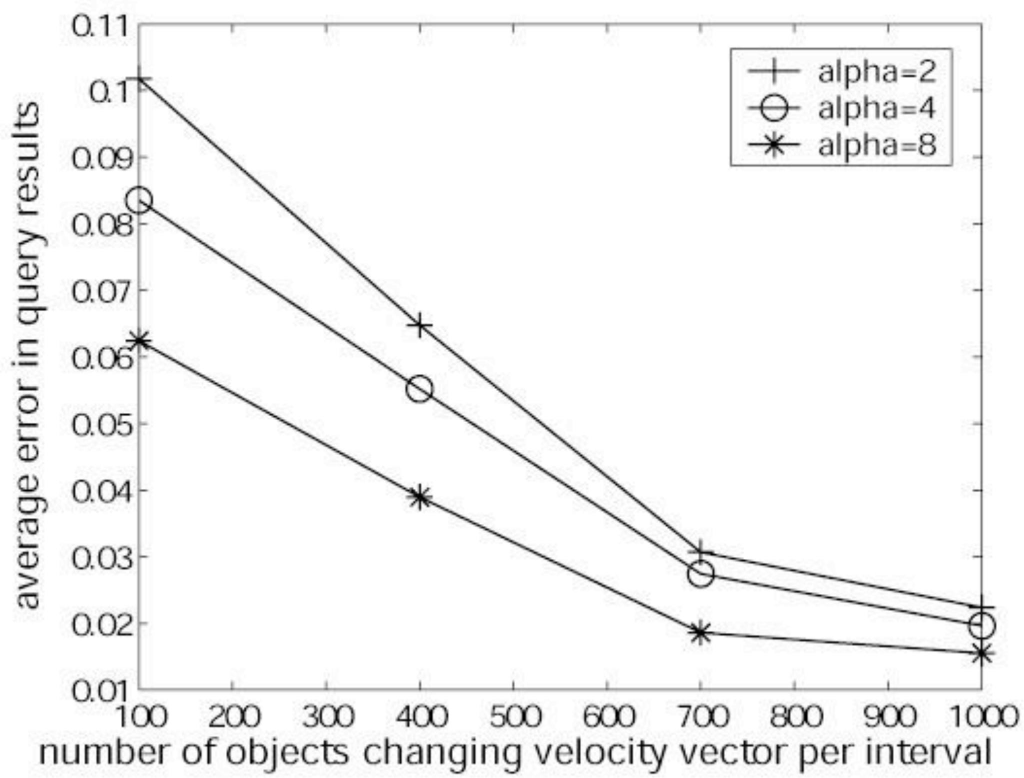
Impact of distributed query processing on server load



Effect of α on server load

Server load in log scale: time spent by the simulation for executing the server side logic per time step

Error Rate with Lazy Propagation



Error associated with lazy query propagation

Query error:

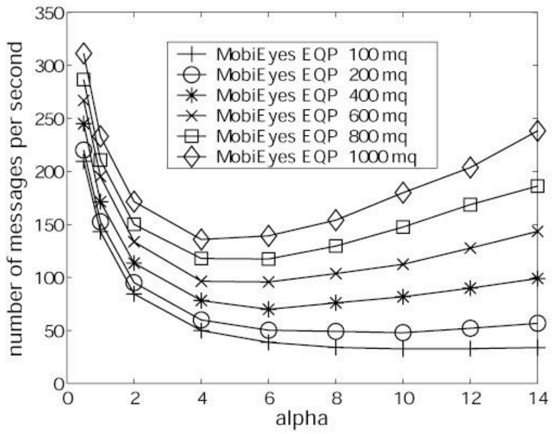
the number of missing objects in the result (compared to the correct result) divided by the size of the correct query result.

Observation:

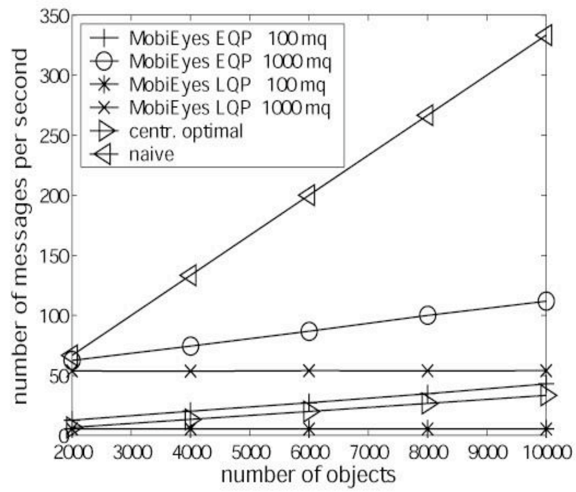
(1) The error in query results decreases with increasing number of objects changing velocity vectors per time step.

(2) Frequent grid cell crossings are expected to decrease the accuracy of the query results.

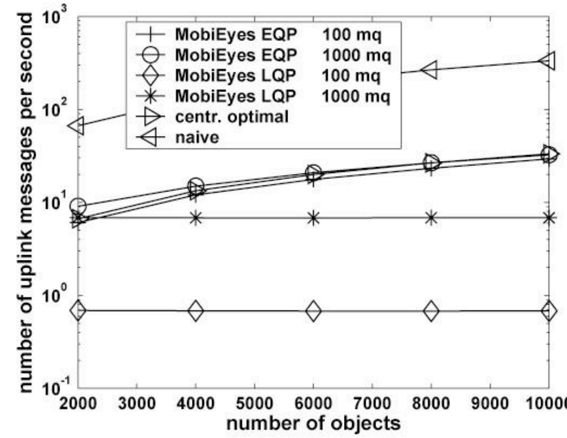
Messaging cost



Effect of α on message costs



Effect of number of objects on messaging cost



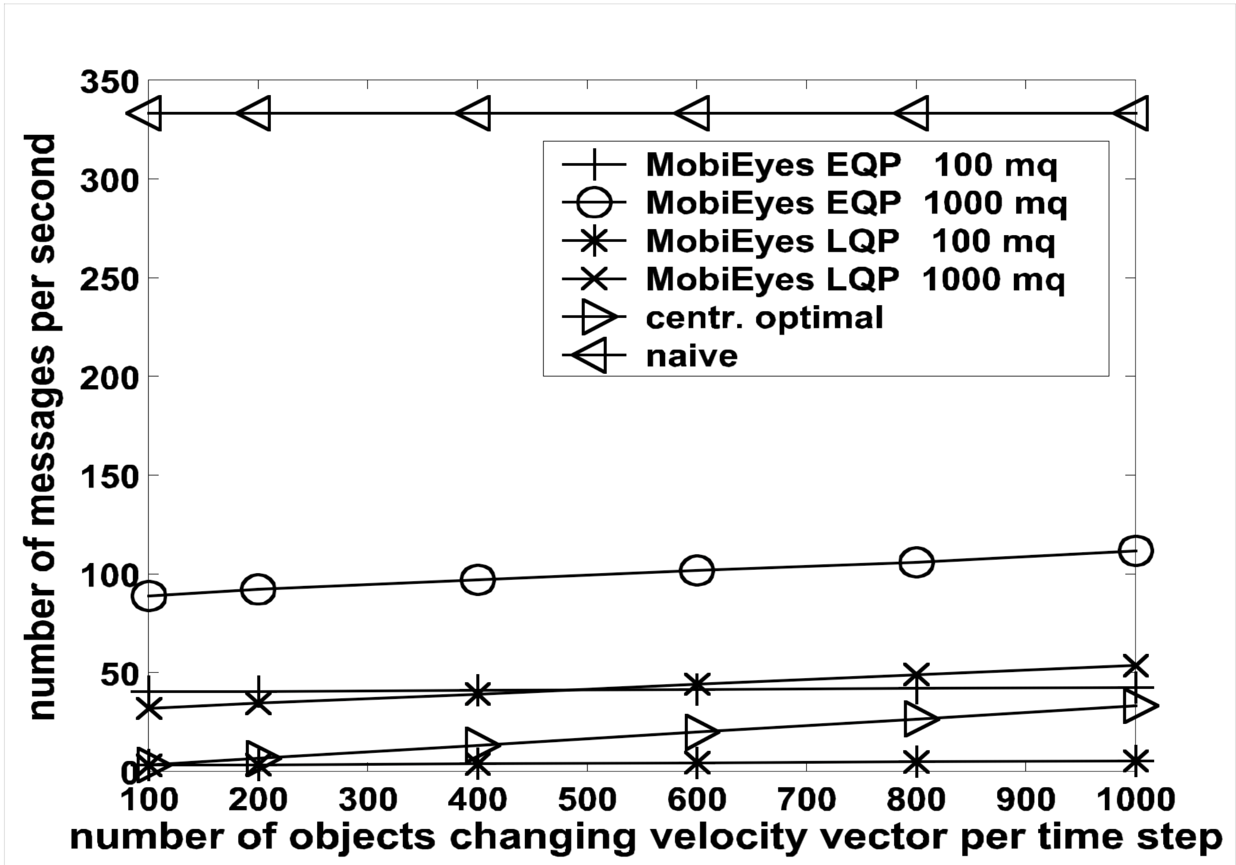
Effect of number of objects on uplink messaging cost

Messaging cost: total number of messages sent on the wireless medium per second (uplink and downlink)

Centralized naïve: send your position when it changes

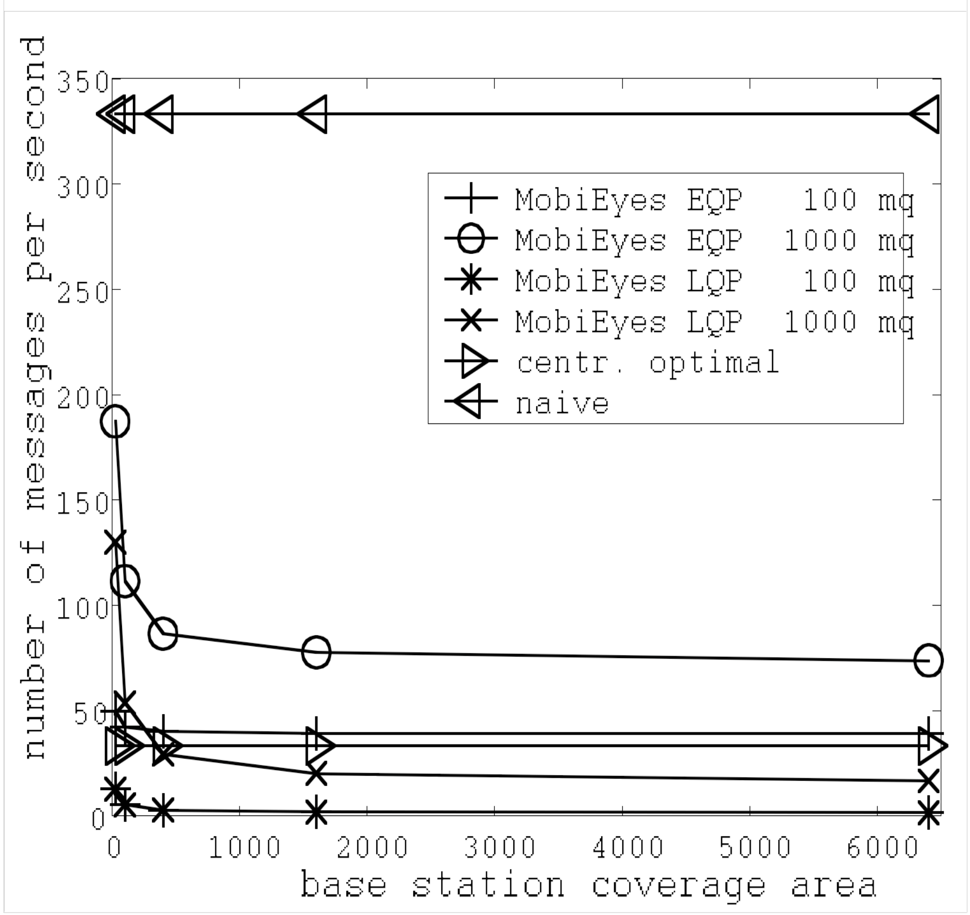
Centralized optimal: send your velocity vector when it changes

Messaging cost



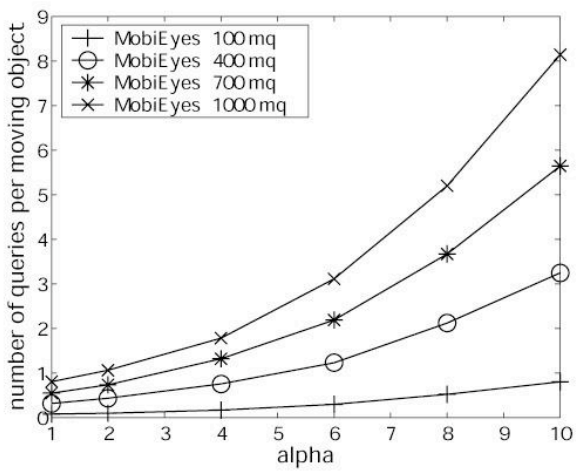
Effect of number of objects changing velocity vector per time step on messaging cost

Messaging cost

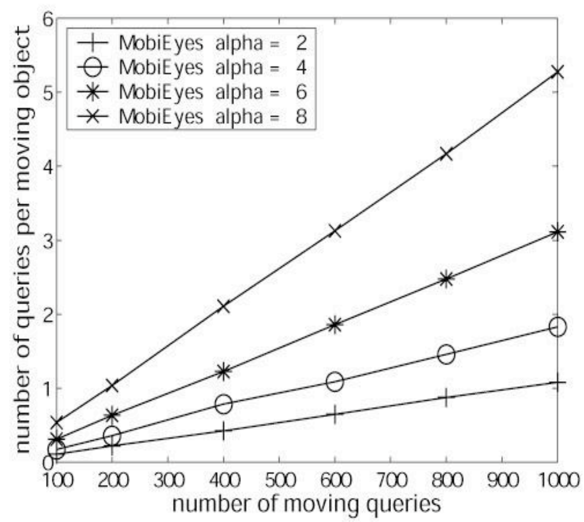


Effect of base station coverage area on messaging cost

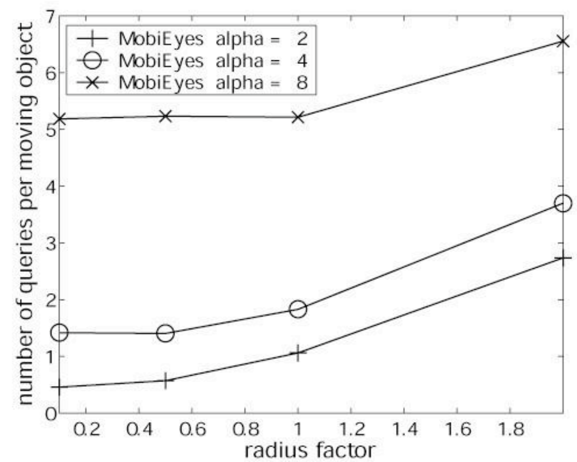
Amount of processing on moving objects



Effect of α on the average number of queries evaluated per step on a moving object



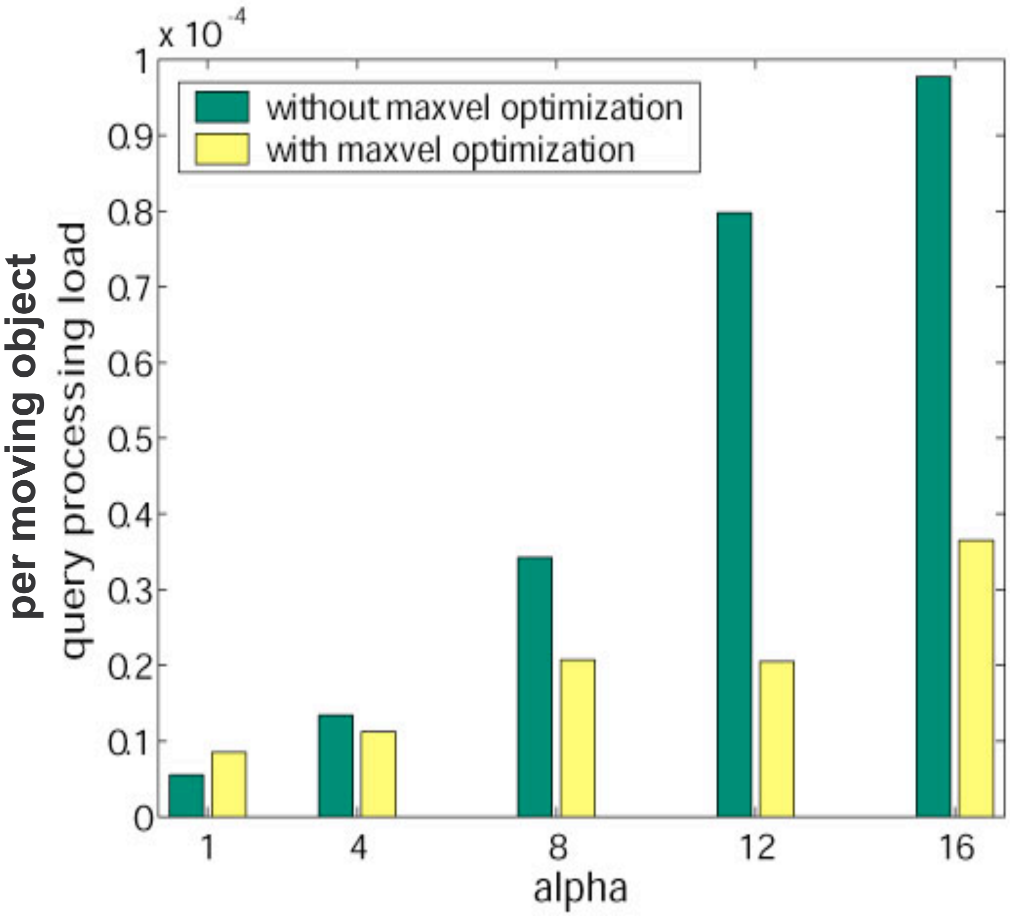
Effect of the total number of queries on the average number of queries evaluated per step on a moving object



Effect of the query radius on the average number of queries evaluated per step on a moving object

Amount of processing on moving objects: number of queries a moving object has to evaluate at each time step

Optimizations: Safe Periods

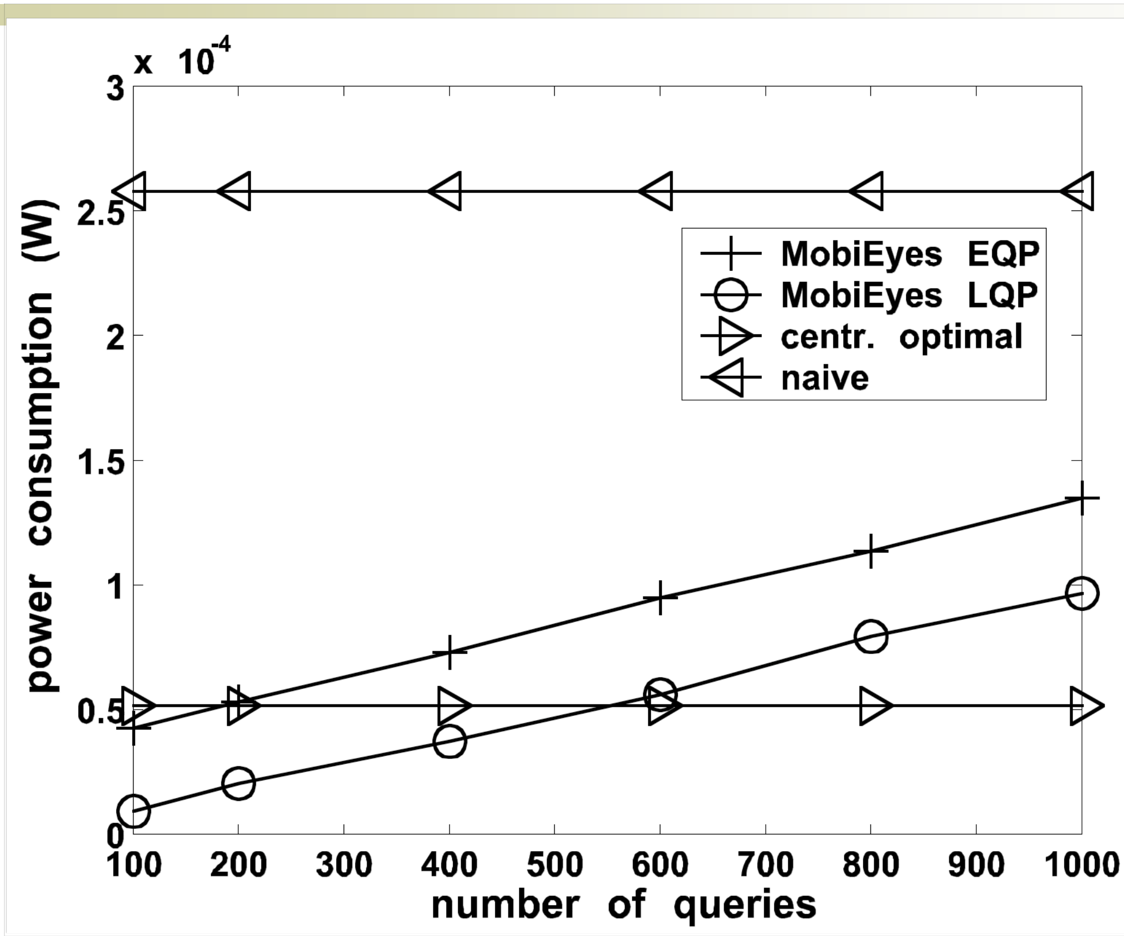


Effect of the safe period optimization on the average query processing load of a moving object

For large values of alpha, the safe period optimization is very effective.

For small alpha value, the safe period optimization incurs a small overhead.

Power Consumption



Effect of number of queries on per object power consumption due to communication

MobiEyes Architecture: Summary

- A distributed approach to location monitoring of moving objects:
  - aiming at reducing both *the server load* and *the network bandwidth* requirements for continuous reporting of location changes to the server
- Technology push:
  - Storage/computing power growth + Wireless connectivity growth
- Locality-based approach
  - Moving location queries are location dependent – high and changing locality
    - Given a set of active location queries, only those mobile objects that are in the geographical vicinity of the focal objects are relevant.

MobiEyes:
Protecting Location Privacy

- Policy-based Location Privacy
- ➡ Anonymization-based Location Privacy

Location Privacy Threats: Examples

- Observation identification
  - if *external observation* is available, it can be used to link a request to an identity
- Restricted space identification
  - a known *location owned by identity* can link a request to an identity
- Precise location tracking
  - *successive position updates* can be linked together even if identifiers are removed from updates

[Beresford et al. 2003], [Gruteser 2003]

Privacy and Anonymity in General

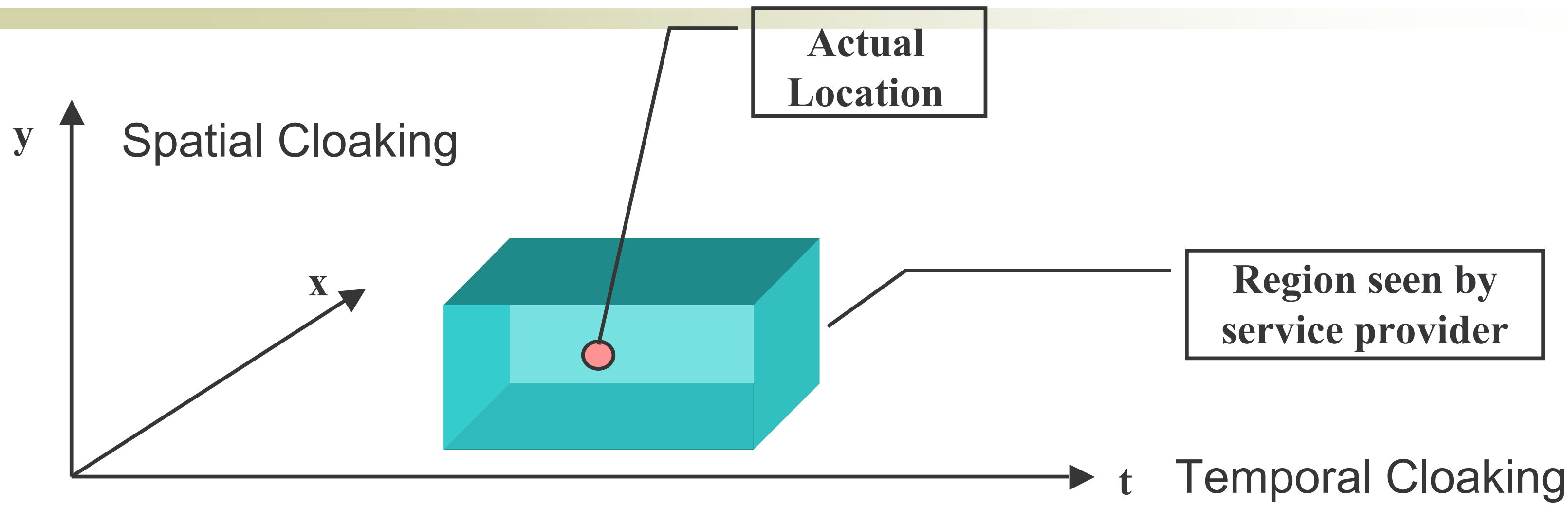
- Privacy: [Beresford et al. 2003)]
  - "The **right/claim of individuals, groups and institutions** to determine for themselves, when, how and to what extent information about them is communicated to others"
- Anonymity:
  - "A **system property** which guarantees that disclosure of information, that leads to the identification of the end users, is prevented."

Location Anonymity using Location Cloaking (Perturbation)

- Protect location privacy by location perturbation
  - Introduce uncertainty on exact location (e.g., location k-anonymity)
  - Example 1:
    - In E-911, handset users are required to be located with an accuracy of 50 to 150 meters
  - Example 2:
    - **Temporal cloaking:**
      - Find taxi nearby within 1 minute → find taxi nearby within 5 minutes
    - **Spatial cloaking**:
      - find taxi within 1 mile of me → find taxi within 5 miles of me
  - More uncertainty → higher k, larger cloaking box → higher privacy of location

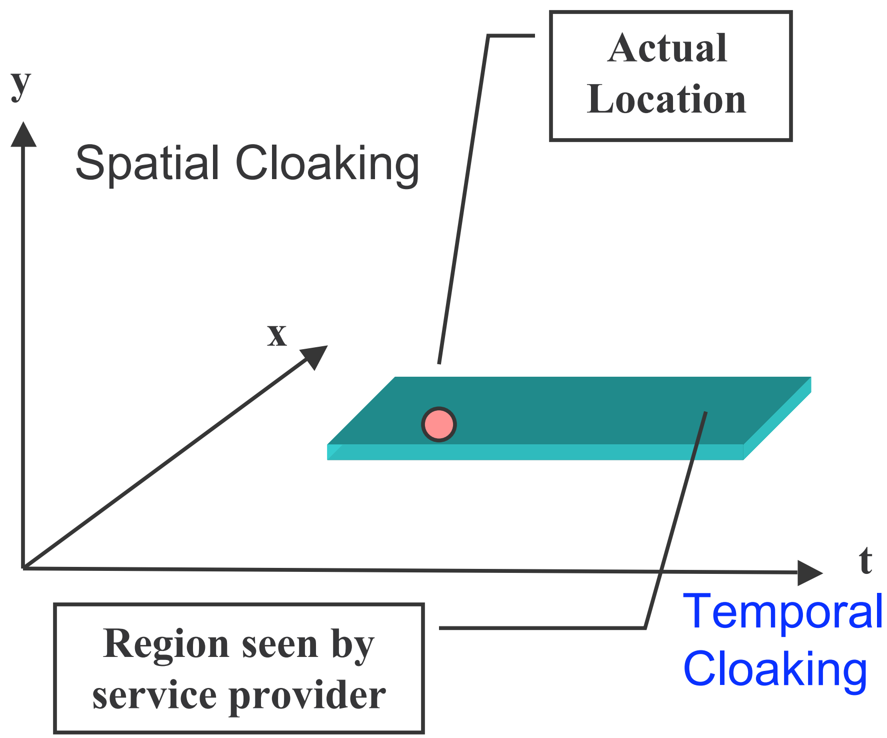
- Tradeoff: Location Privacy v.s. Location Service Quality
  - More ambiguous location information may lead to certain degradation in the quality of the service
- Technical Challenge
  - How to balance location privacy and location service quality?

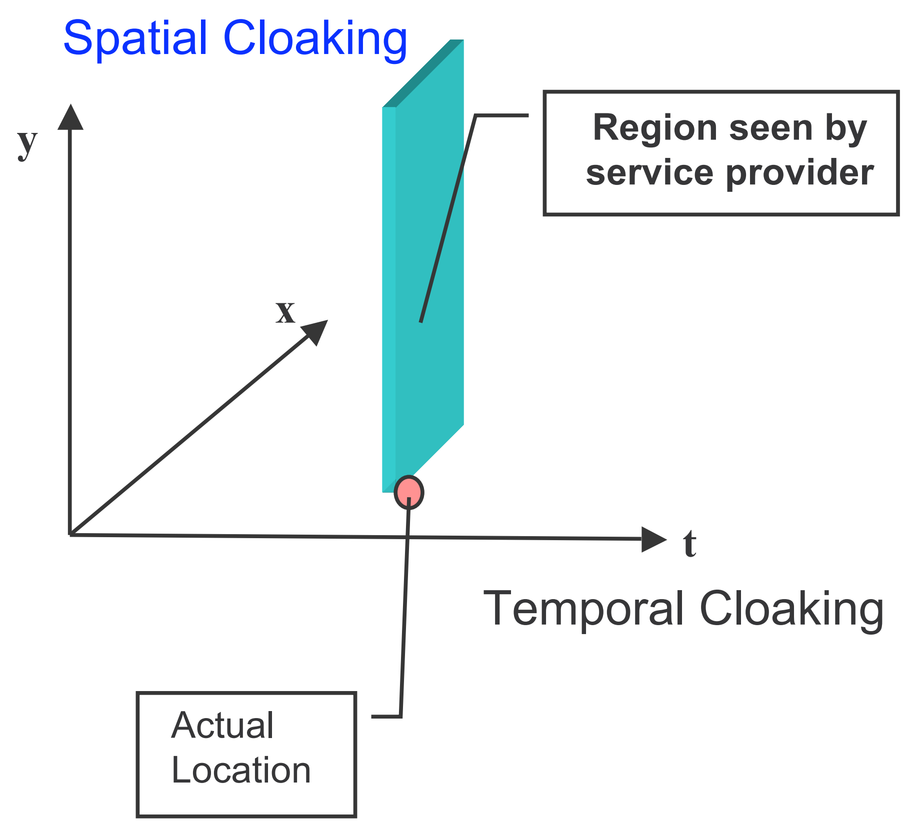
Privacy Requirements



- Privacy Requirements serve as constraints for location cloaking
  - *Location k*-anonymity (Beresford 03, Gruteser 03):
    - At least *k* users inside the region such as a circle of radius *r*
  - *Spatial location uncertainty tolerance*
  - *Temporal location uncertainty tolerance*
- **Example:**
  - find taxi within 5 mile of me right now with spatial tolerance of 2 miles, temporal tolerance of next 5 minutes, and k-anonymity of k=5.

Spatial or Temporal Cloaking of Location



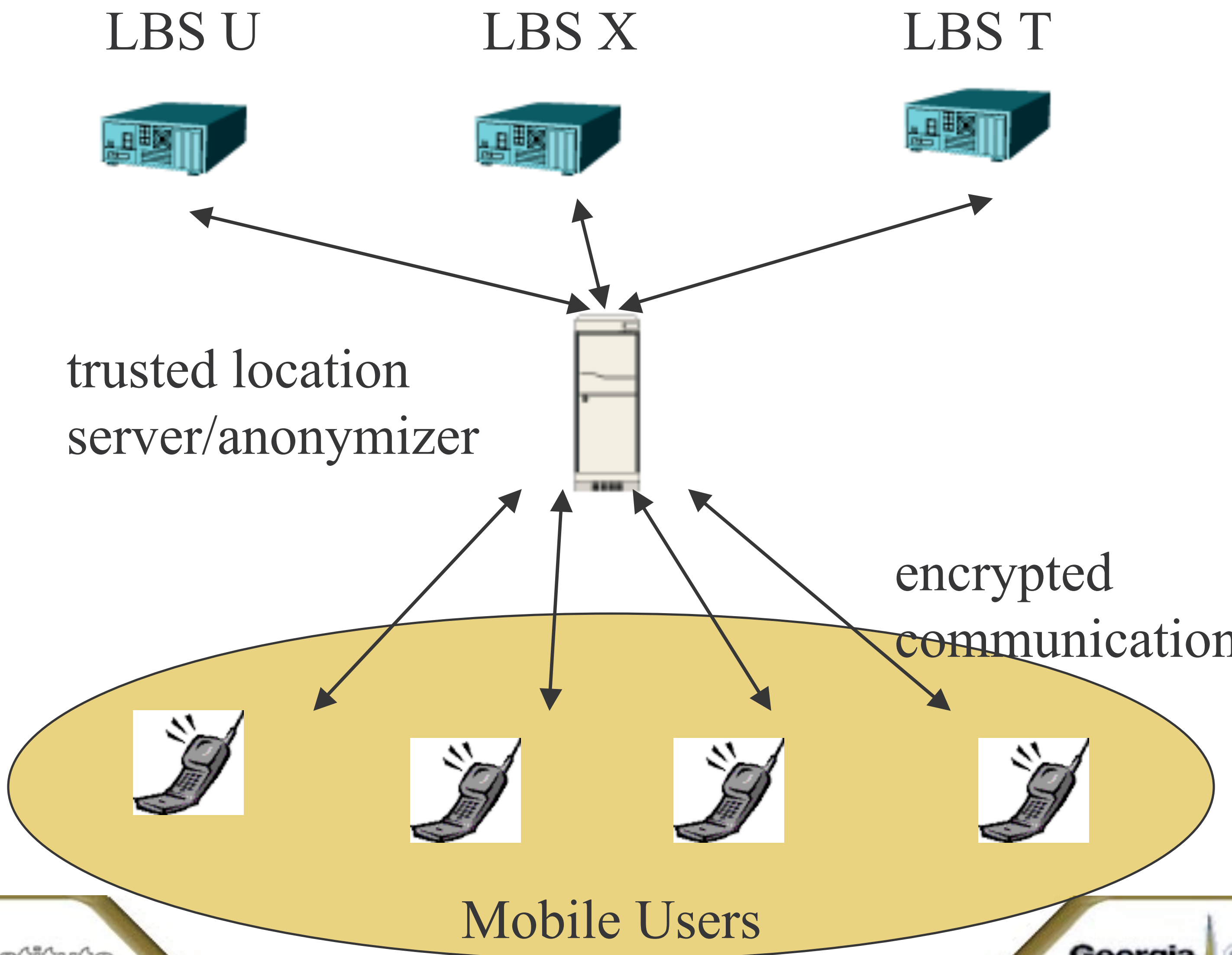


MobiEyes' Location Privacy Solution

- Introduce a personalized *k*-anonymity model. Each message can specify:
  - a different *k* value based on its specific privacy requirement
  - spatial and temporal tolerance values based on its QoS requirements
- Develop a cloaking algorithm, called *CliqueCloak*, capable of
  - supporting customizable location *k*-anonymity model
  - continuously processing a stream of messages

MobiEyes's Location Cloaking System Architecture

[Gedik and Liu icdcs 2005]

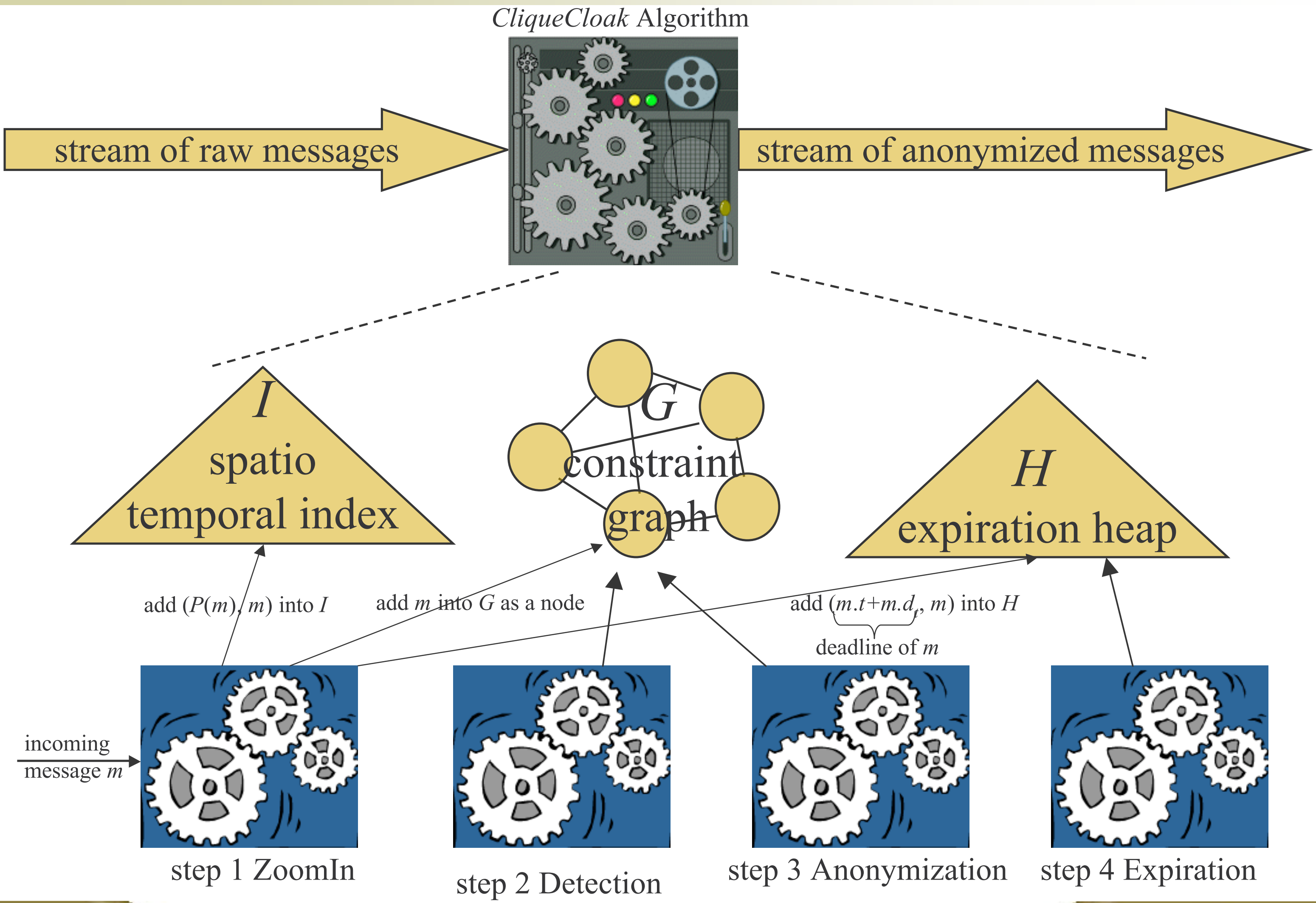


Location Cloaking: The Road Map

Upon arrival of a LBS service request message

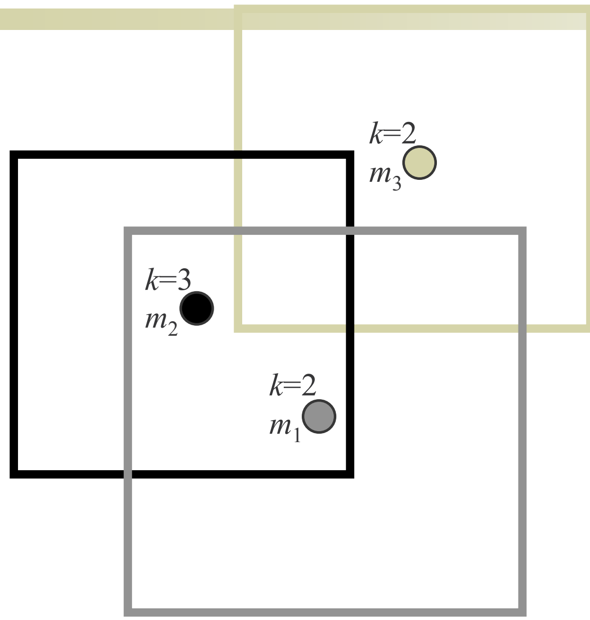
- Perturb the message based on user's QoS specification
- Location k-anonymization
  - Check message queue
    - If there are k-1 other messages in the same message constrain box as the new message, anonymize the k messages together and send them to the service provider
    - Otherwise, insert the new message in the message queue, and wait for the next new message
  - Goal
    - Anonymize as many messages as possible – reduce dropped service requests due to k-location anonymity requirement
  - Challenges
    - Variable k
    - Constraint box: Temporal and Spatial Location Uncertainty Tolerance

Location Cloaking Engine

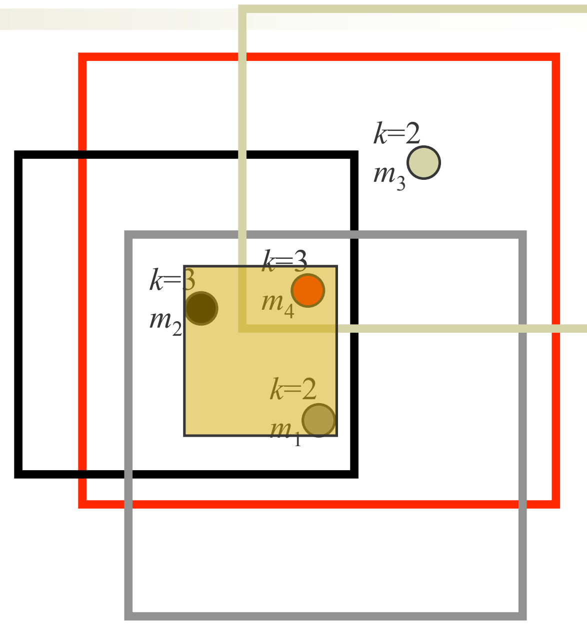


Georgia Institute of Technology
Georgia Tech College of Computing

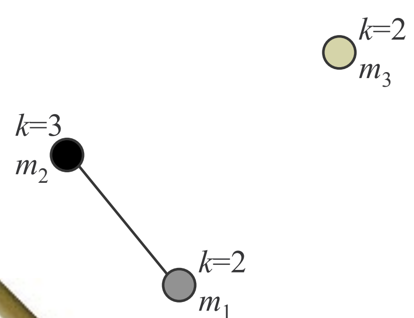
Illustration of *CC* Theorem



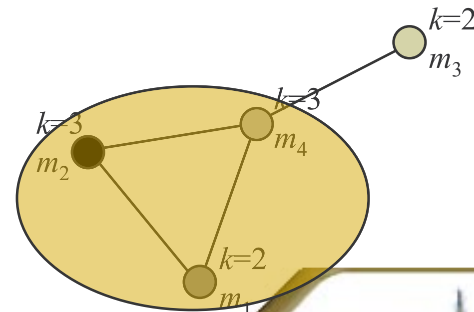
spatial layout I



spatial layout II



constraint graph I



constraint graph I

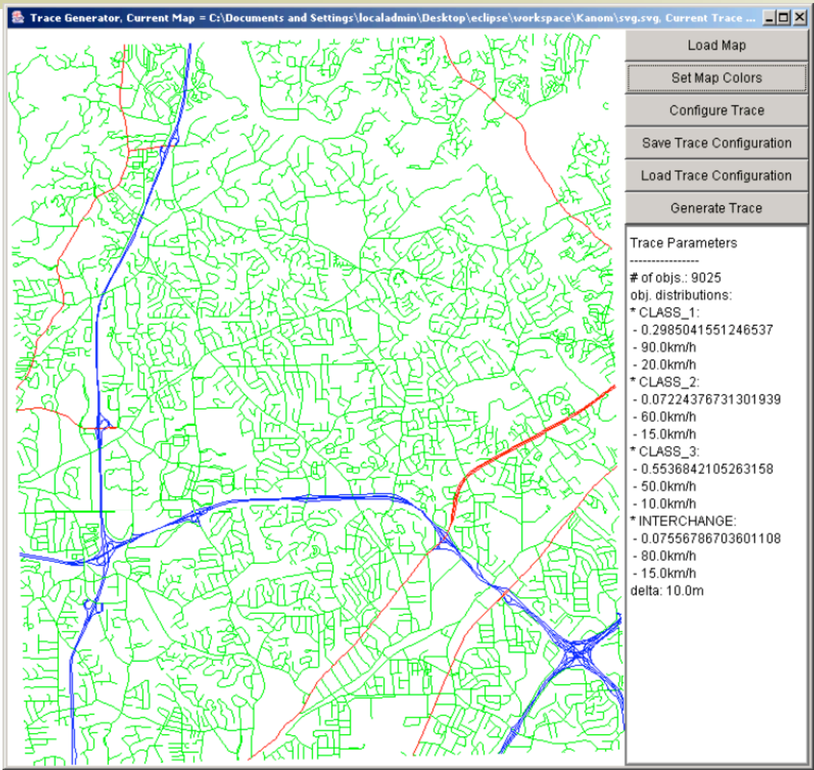
Local-*k* vs. Nbr-*k* Search

- During the clique search phase, let *m* be the new message, we can search for a
  - clique of size *m*.*k*
  - this is called local-*k* search
- Or we can try to maximize the size of the clique by iterating on the list
  - $sort_{desc}$ $\{k: k \leq 1+|nbr(m)|,\ k=m'.k,\ m' \in nbr(m)\}$
  - this is called nbr-*k* search

Deferred vs. Immediate Search

- Do we have to search for a clique every time we receive a new message?
- Yes → immediate search
- No → deferred search: we check $|nbr(m)| > \alpha * m.k$
  - If satisfied perform search now
  - Otherwise defer it: If not picked up until its deadline, perform search

Experimental Setup



trace generator

mean of car speeds for each road type	$\{90, 60, 50\} km/h$
std.dev. in car speeds for each road type	$\{20, 15, 10\} km/h$
traffic volume data	$\{2916.6, 916.6, 250\}$ per hour

car movement parameters

- Road data available from United States Geological Survey (USGS) in SDTS format
- Use transportation layer of 1:24K Digital Line Graphs (DLGs).
- Extract three types of roads
  - class 1 (expressway)
  - class 2 (arterial)
  - class 3 (collector)
- Map from Chamblee region of Georgia
- Covers a region of ≈ 160km2
- Use real traffic volume data to calculate the number of cars on each road type
- Simulate cars moving on roads
- The trace has a duration of one hour

Experimental Parameters

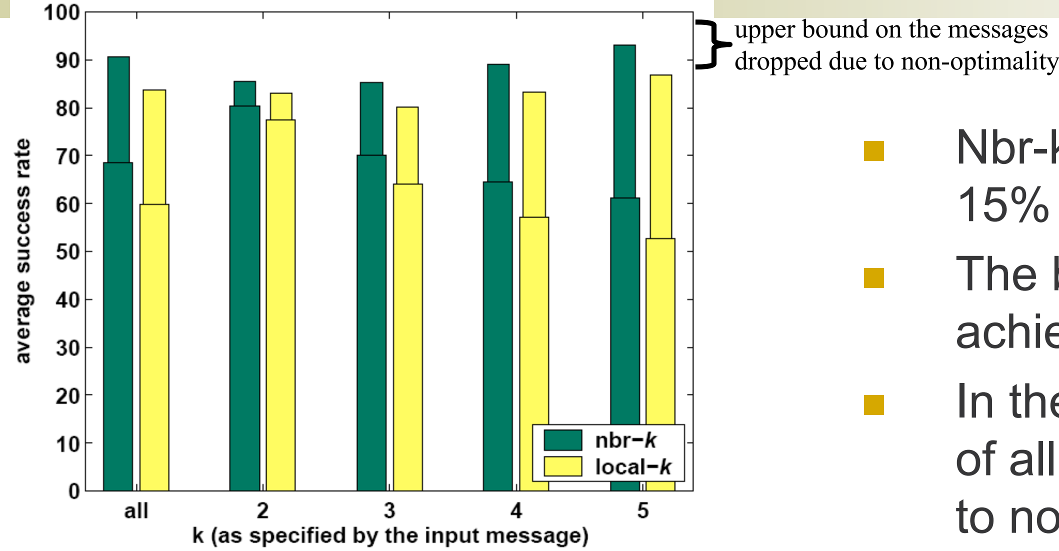
- Each car generates several messages during the simulation.
- Each message specifies an anonymity level (*k* value) from the list
  {5, 4, 3, 2} using a Zipf parameter of 0.6
- The spatial and temporal tolerance values of the messages are selected independently using normal distributions
- Whenever a message is generated, the originator of the message waits until the message is anonymized or dropped, after which it waits for a normally distributed amount of time, called the *inter-wait time*

Parameter	Default value
anonymity level range	{5, 4, 3, 2}
anonymity level zipf param	0.6
mean spatial tolerance	$100m$
variance in spatial tolerance	$40m^2$
mean temporal tolerance	$30s$
variance in temporal tolerance	$12s^2$
mean inter-wait time	$15s$
variance in inter-wait time	$6s^2$

message generation parameters

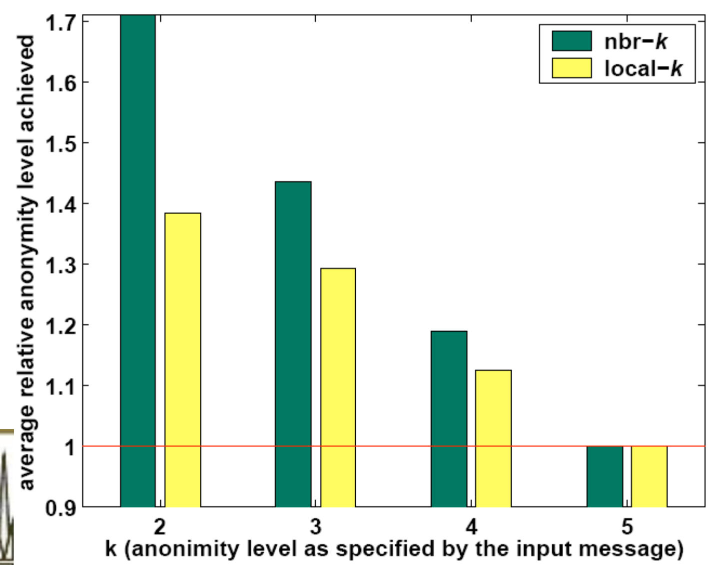
Experimental Results

Success rates for different *k* values



- Nbr-k approach provides around 15% better average success rate
- The best average success rate achieved is around 70
- In the worst case remaining 10% of all messages are dropped due to non-optimality of the algorithm

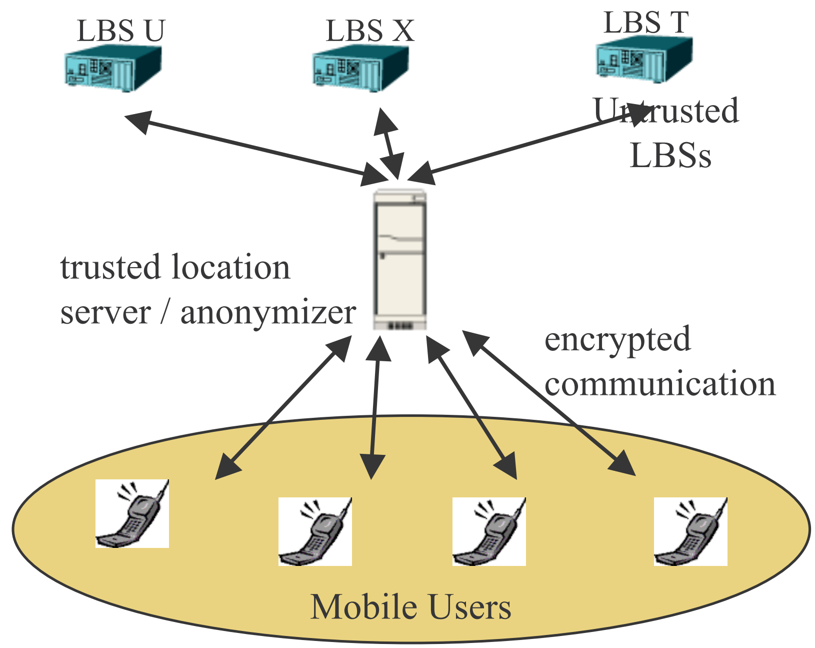
Relative anonymity levels for different *k* values



- Nbr-*k* shows a relative anonymity level of 1.7 for messages with *k* = 2
- Local-*k* shows a lower relative anonymity level of 1.4 for messages with *k* = 2
- The gap vanishes for *k* = 5

Scalable Location Anonymity for Continuous use of LBSs

- Solution:
  - *Spatio-temporal cloaking*
  - *Personalized location k-anonymity*
    - Support for users with different privacy requirements
    - Adjustable QoS / performance tradeoffs
- Ongoing Work
  - Decentralized solutions



Questions

www.cc.gatech.edu/disl/projects/Mobieyes/